CLAIM AMENDMENTS:

Listing of the Claims:

1-43. (Cancelled)

44. (New) An automated method of detection of software vulnerabilities by applying a rule set to test for vulnerabilities in computer software, the rule set comprising at least one vulnerability characterisation rule, the method incorporating the steps of:

a) providing a training data set of computer software incorporating positive and negative vulnerability examples and expressed as programs flagged to indicate either presence or absence of vulnerability, the programs comprising instructions each incorporating an identifier to indicate its associated program, the instruction's address, an instruction operator and a list of instruction operands,

b) defining a rule generalisation, the rule generalisation being processable to transform it into the at least one vulnerability characterisation rule, and

c) using computer apparatus to execute the steps of:

i) receiving the training data set and the rule generalisation,

ii) processing the rule generalisation to transform it into a more specific rule generalisation by employing logic of at least First-Order and adding to the rule generalisation at least one of a condition, a variable, a constant, a unification of variables and a function based on the training data set and background knowledge relating to attributes of the training data set and consisting of at least one of concepts, facts of interest and functions for calculating values of interest from items of data,

iii) evaluating the more specific rule generalisation by applying it to the training data set to identify vulnerabilities, and

iv) incorporating the more specific rule generalisation in the rule set if it classifies vulnerabilities in the training data set adequately in terms of covering at least some of the positive vulnerability examples,

v) applying the rule set to a test program for vulnerability detection therein, and

vi) providing an alert or a report to a user regarding vulnerability detection in the test program resulting from operation of the method in order to enable corrective action to be taken.

45. (New) An automated method of detection of software vulnerabilities according to claim 44 wherein the background knowledge includes at least one of the following tests:-

a) a copying loop test,

b) a length loop test,

c) a test for one block following another,

d) a test for making a call to a string length function at a specified index,

e) a test for tests that precede conditional jumps in an instruction list,

f) a test for whether or not a given list is empty,

g) a test for whether or not a given list has a single jump test (conditional), and if it is, to return a tested register, and

h) a test for whether or not a given list modifies a given register.

46. (New) An automated method of detection of software vulnerabilities according to claim 44 wherein the rule set incorporates a rule which classifies a program as vulnerable if there is a copying loop preceded by a call to a string length function, with no conditional jumps therebetween.

47. (New) An automated method of detection of software vulnerabilities according to claim 44 wherein the rule set incorporates a rule which classifies a program as vulnerable if there is a copying loop is defined as a portion of code that copies to a register from a source pointer, changes the source pointer, copies from the register into a destination pointer, changes that destination pointer, and has a control flow path from the code portion's end back to the code portion's beginning thus forming a loop.

48. (New) An automated method of detection of software vulnerabilities according to claim 44 wherein the rule set incorporates a rule which classifies a program as vulnerable if

there is a copying loop with a test for zero, and one other test, but a register referenced by the other test is not used during the loop.

49. (New) A system for computer-implemented detection software vulnerabilities by applying a rule set to test for vulnerabilities in computer software, the rule set comprising at least one vulnerability characterisation rule, the system incorporating computer apparatus incorporating a computer program stored in a hardware recording medium and being programmed by such computer program to carry out the steps of:

a) receiving a training data set of computer software incorporating positive and negative vulnerability examples and expressed as programs flagged to indicate either presence or absence of vulnerability, the programs comprising instructions each incorporating an identifier to indicate its associated program, the instruction's address, an instruction operator and a list of instruction operands,

b) receiving a rule generalisation which is processable to transform it into the at least one vulnerability characterisation rule, and

c) processing the rule generalisation to transform it into a more specific rule generalisation by employing logic of at least First-Order and adding to the rule generalisation at least one of a condition, a variable, a constant, a unification of variables and a function based on the training data set and background knowledge relating to attributes of the training data set and consisting of at least one of concepts, facts of interest and functions for calculating values of interest from items of data,

d) evaluating the more specific rule generalisation by applying it to the training data set to identify vulnerabilities, and

e) incorporating the more specific rule generalisation in the rule set if it classifies vulnerabilities in the training data set adequately in terms of covering at least some of the positive vulnerability examples,

f) applying the rule set to a test program for vulnerability detection therein, and

g) providing an alert or a report to a user regarding vulnerability detection in the test program resulting from operation of the method in order to enable corrective action to be taken.

50. (New) A system for detection of software vulnerabilities according to claim 49 wherein the background knowledge includes at least one of the following tests:-

a) a copying loop test,

b) a length loop test,

c) a test for one block following another,

d) a test for making a call to a string length function at a specified index,

e) a test for tests that precede conditional jumps in an instruction list,

f) a test for whether or not a given list is empty,

g) a test for whether or not a given list has a single jump test (conditional), and if it is, to return a tested register, and

h) a test for whether or not a given list modifies a given register.

51. (New) A system for detection of software vulnerabilities according to claim 49 wherein the rule set incorporates a rule which classifies a program as vulnerable if there is a copying loop preceded by a call to a string length function, with no conditional jumps therebetween.

52. (New) A system for detection of software vulnerabilities according to claim 49 wherein the rule set incorporates a rule which classifies a program as vulnerable if there is a copying loop is defined as a portion of code that copies to a register from a source pointer, changes the source pointer, copies from the register into a destination pointer, changes that destination pointer, and has a control flow path from the code portion's end back to the code portion's beginning thus forming a loop.

53. (New) A system for detection of software vulnerabilities according to claim 49 wherein the rule set incorporates a rule which classifies a program as vulnerable if there is a

copying loop with a test for zero, and one other test, but a register referenced by the other test is not used during the loop.

54. (New) A computer readable hardware medium which embodies computer readable instructions for controlling operation of computer apparatus to implement detection of software vulnerabilities by applying a rule set to test for vulnerabilities in computer software, the rule set comprising at least one vulnerability characterisation rule, wherein the instructions provide for control of the computer apparatus to carry out the steps of:

a) receiving a training data set of computer software incorporating positive and negative vulnerability examples and expressed as programs flagged to indicate either presence or absence of vulnerability, the programs comprising instructions each incorporating an identifier to indicate its associated program, the instruction's address, an instruction operator and a list of instruction operands,

b) receiving a rule generalisation which is processable to transform it into the at least one vulnerability characterisation rule, and

c) processing the rule generalisation to transform it into a more specific rule generalisation by employing logic of at least First-Order and adding to the rule generalisation at least one of a condition, a variable, a constant, a unification of variables and a function based on the training data set and background knowledge relating to attributes of the training data set and consisting of at least one of concepts, facts of interest and functions for calculating values of interest from items of data,

d) evaluating the more specific rule generalisation by applying it to the training data set to identify vulnerabilities, and

e) incorporating the more specific rule generalisation in the rule set if it classifies vulnerabilities in the training data set adequately in terms of covering at least some of the positive vulnerability examples,

f) applying the rule set to a test program for vulnerability detection therein, and

g) providing an alert or a report to a user regarding vulnerability detection in the test program resulting from operation of the method in order to enable corrective action to be taken.

55. (New) A computer readable hardware medium according to claim 54 wherein the background knowledge includes at least one of the following tests:-

a) a copying loop test,

b) a length loop test,

c) a test for one block following another,

d) a test for making a call to a string length function at a specified index,

e) a test for tests that precede conditional jumps in an instruction list,

f) a test for whether or not a given list is empty,

g) a test for whether or not a given list has a single jump test (conditional), and if it is, to return a tested register, and

h) a test for whether or not a given list modifies a given register.

56. (New) A computer readable hardware medium according to claim 54 wherein the rule set incorporates a rule which classifies a program as vulnerable if there is a copying loop preceded by a call to a string length function, with no conditional jumps therebetween.

57. (New) A computer readable hardware medium according to claim 54 wherein the rule set incorporates a rule which classifies a program as vulnerable if there is a copying loop is defined as a portion of code that copies to a register from a source pointer, changes the source pointer, copies from the register into a destination pointer, changes that destination pointer, and has a control flow path from the code portion's end back to the code portion's beginning thus forming a loop.

58. (New) A computer readable hardware medium according to claim 54 wherein the rule set incorporates a rule which classifies a program as vulnerable if there is a copying loop

with a test for zero, and one other test, but a register referenced by the other test is not used during the loop.